

BUNDESREPUBLIK DEUTSCHLAND

003301

EP04/330119.04.2004



REC'D 10 MAY 2004	
WIPO	PCT

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen: 103 14 834.5

Anmeldetag: 01. April 2003

Anmelder/Inhaber: Siemens Aktiengesellschaft,
80333 München/DE

Bezeichnung: Verfahren Anordnung zur Modifikation von Quell-
code unter Einbeziehung zusätzlicher Informationen

IPC: G 06 F 17/50

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 15. April 2004
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag

Sieck

**PRIORITY
DOCUMENT**
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH RULE 17.1(a) OR (b)

Beschreibung

Verfahren Anordnung zur Modifikation von Quellcode unter
Einbeziehung zusätzlicher Informationen

5

Die Erfindung betrifft ein Verfahren und eine Anordnung zur
Modifikation von Quellcode, bei dem/der ein Quellcode,
beispielsweise ein Java- Quellcode, in eine Darstellung in
einer Meta-Auszeichnungssprache, beispielsweise XML,
10 übergeführt, dort, beispielsweise mit XSLT, transformiert und
dann diese in der Meta-Auszeichnungssprache formulierte
transformierte Darstellung in einen modifizierten Quellcode,
beispielsweise derselben Ausgangssprache, zurückverwandelt
wird.

15

Aus dem Internet ist unter <http://beautyj.berlios.de/> ein
Java Source Code Transformation Tool BeautyJ bekannt, bei dem
ein Java Quellcode in eine XML-Darstellung umgewandelt wird,
mittels Sourclet API, beispielsweise durch Einfügen von
20 Leerzeichen oder geänderten Kommentaren an bestimmten
Stellen, „verschönert“ und anschließend der modifizierte
Quellcode in Java Quellcode zurück konvertiert werden kann.
Eine Transformation mittels XSLT wird hier, für diesen Zweck,
nur vorgeschlagen, aber nicht umgesetzt.

20

Die der Erfindung zugrunde liegende Aufgabe liegt nun darin,
ein Verfahren und eine Anordnung zur Modifikation von
Quellcode anzugeben, bei dem/der eine weitergehende noch
flexiblere und effizientere Modifikation der Quellcodes
30 erreicht wird.

30

Diese Aufgabe wird hinsichtlich des Verfahrens durch die
Merkmale des Patentanspruchs 1 und hinsichtlich der
Anordnung durch die Merkmale des Anspruchs 11 erfindungsgemäß
35 gelöst. Die weiteren Ansprüche betreffen bevorzugte
Ausgestaltungen der Erfindung.

35

Die Erfindung besteht im Wesentlichen darin, dass der in eine Meta-Auszeichnungssprache, beispielsweise XML, transformierte Quellcode mit weiteren Informationen, wie beispielsweise Ausgangszuständen, auszutauschenden Code-Fragmenten und auf die jeweilige natürliche Sprache des Anwenders zugeschnittenen Fremdsprachenmodulen, mit einer in seinen Elementen standardisierten und übersichtlich beschreibbaren Transformation, beispielsweise XSLT, vermischt wird, wodurch, nach einer Rückkonvertierung aus XML in die ursprüngliche Programmiersprache, ein neuer Quellcode entsteht, bei dem nicht nur die Darstellung, sondern auch der eigentliche Programminhalt bzw. der Programmablauf entsprechend den weiteren Informationen und den Transformationsvorschriften verändert wurde.

Die Erfindung wird im Folgenden anhand von in den Zeichnungen dargestellten Beispielen näher erläutert. Dabei zeigt

Zeichnung 1 ein Gesamtblockdiagramm zur Erläuterung der Erfindung,

Zeichnung 2 ein Blockschaltbild zur Erläuterung des erfindungsgemäßen Austauschs von Code-Fragmenten,

Zeichnung 3 ein Blockschaltbild zur Erläuterung des erfindungsgemäßen Einfügens von Zustandsdaten,

Zeichnung 4 ein Blockschaltbild zur Erläuterung der Variationsmöglichkeiten des erfindungsgemäßen Einbaus von Informationen und

Zeichnung 5 ein Blockschaltbild zur Erläuterung des erfindungsgemäßen Einbaus von Fremdsprachmodulen zur Internationalisierung des Quellcodes.

In **Zeichnung 1** ist ein Gesamtblockdiagramm zur Erläuterung der Erfindung dargestellt, bei dem zunächst ein Quellcode SC

durch einen Konverter **CONV** in einen in einer Meta-Auszeichnungssprache formulierten ersten Code **CodeML** umwandelt wird, wobei der Quellcode **SC** bei sofortiger Kompilierung einen Byte Code bzw. Binärcode **B** ergeben kann.

5 Zu dem in der Meta-Auszeichnungssprache dargestellten Code **CodeML** wird nun eine zusätzliche Information **INFO** auf dem Wege einer durch Transformationsregeln **TR** beschriebenen Transformation **T** dem Code **CodeML** bzw. dem letztlich dem Quellcode **SC** hinzugefügt, wodurch sich ein zweiter ebenfalls

10 in der Meta-Auszeichnungssprache formulierten Code **CodeML*** ergibt. Ein weiterer Konverter **RCONV** wandelt nach der Transformation den Code **CodeML*** in einen Quellcode **SC*** zurück, der typischerweise in derselben Sprache wie der Quellcode **SC** formuliert ist. Durch einen Compiler **COMP** wird

15 schließlich der modifizierte Code **SC*** in einen modifizierten Byte Code **B*** oder aber gleich in einen ausführbaren Binärcode umgewandelt. Wesentlich ist hierbei, dass sich der Byte-Code **B*** vom Byte-Code **B** unterscheidet bzw. dass der Quellcode nicht nur in seiner Darstellung, sondern auch in seinem

20 Programmablauf geändert wurde.

Der Quellcode **SC** und der modifizierte Quellcode **SC*** sind beispielsweise in der Programmiersprache Java und die Codes **CodeML** und **CodeML*** sind beispielsweise in der Meta-Auszeichnungssprache XML formuliert, wobei „XML“ für Extended Markup Language steht.

In **Zeichnung 3** ist ein erstes Ausführungsbeispiel dargestellt, bei dem die zusätzliche Information **INFO** in Form

30 von Daten **D**, bspw. Initialisierungszuständen **SSDb**, Zustandsdaten **SDa**, Datenbankdaten **Dc**, beliebige Daten **x**, dem **CodeML** hinzugemischt werden, wobei diese Daten beispielsweise feste Zustände bzw. Werte für Konstanten und Variablen darstellen. Auf diese Weise kann der Quellcode **SC** mit festen

35 Zuständen versorgt und transformiert werden, so dass ein benötigter Zustand zur Laufzeit des Programms, z.B. als Initialisierungszustand **SSDb**, sofort zur Verfügung steht und

nicht mehr gesondert ermittelt werden muss. Auf diese Weise können auch Objekt-Zustände in den Code eingebracht werden, die ein Wiederaufsetzen (Recovery) eines unterbrochenen Programms am selben Ort mit denselben Zuständen ermöglichen, ohne dass zusätzliche aufwändige programmtechnische Vorkehrungen hierfür getroffen werden müssen.

Die Transformation **T**, z. B. eine Extended Stylesheet Language Transformation oder **XSLT**, wird durch Transformationsregeln **TR**, z. B. innerhalb von **XSL** (Extended Stylesheet Language) Dateien beschrieben, wobei bspw. die in XSL formulierten Regeln u.a. beschreiben wie der in XML-codierte Quellcode **CodeML** mit den Zustandsdaten aus **D** kombiniert wird, um einen neuen modifizierten Quellcode **CodeML*** mit **SSDb**, **SDa** und **Dc** zu bilden.

Die Regeln einer Transformation **T** können dabei so geartet sein, dass die Informationen in ihrer ursprünglichen Form aber auch in einer durch Regeln geänderten Form hinzugemischt werden.

Die Regeln einer Transformation **T** können dabei auch so geartet sein, dass die Transformation **T**, z.B. mit Hilfe von if-conditions, durch die Informationen bzw. Daten beeinflusst werden.

Die im **Anhang 1** befindlichen Programmauflistungen **Listing 1** bis **Listing 6** zeigen dies an einem konkreten Beispiel, bei dem in einer Testklasse des Quellcodes die nicht initialisierte Variable **String m_sWelcome** in eine initialisierte Form **String m_sWelcome = "hello";** transformiert wird. Hierbei zeigt das **Listing 1** das entsprechende Java-Programm **TestOutput.java**, dass in eine XML-Darstellung **TestOutput.xjava** konvertiert wird. In **Listing 3** ist die XML-Datei **State.XML** mit dem Zustandswert **"hello"** dargestellt. In **Listing 4** folgt dann die Transformationsanweisung zum Vermischen **Mixing.xsl**, die mit

Anweisungen wie **template match =** und **apply-templates** dafür sorgen, dass der Code an der richtigen Stelle modifiziert wird.

- 5 In **Zeichnung 2** ist ein zweites Ausführungsbeispiel dargestellt, bei dem ein in XML codiertes Code-Fragment **CFb** mit einem ursprünglichen in XML codierten Quellcode **CodeML**, der ein Code-Fragment **CFa** enthält, durch die Transformation **T** derart transformiert wird, dass im modifizierten XML-
- 10 codierten Quellcode **CodeML*** anstelle des bisher vorhandenen Fragments **CFa** ein Code-Fragment **CFb** enthalten ist. Die Transformation **T** wird hierbei auch von Transformationsregeln **TR** gesteuert. Ein derartiger Austausch von Code-Fragmenten kann unter bestimmten Umständen z.B. als „Patching“
- 15 bezeichnet werden. Durch das erfindungsgemäße Verfahren kann ein Patching in konsistenter Weise mit einem maximalen Grad an Freiheit für den Softwareentwickler erreicht werden, wobei dies beispielsweise automatisch und unter Berücksichtigung gegenseitiger Abhängigkeiten erfolgen kann.
- 20


Für dieses Ausführungsbeispiel ist ein konkreter Fall durch die **Listings 1A bis 6A** der im **Anhang 2** befindlichen Programmauflistungen gezeigt. Aus dem Java-Source-Code **TestOutput.java** wird wiederum ein **TextOutput.xjava** erzeugt. In **Listing 3A** kan man die Datei **CodeFragment.xml** erkennen, die ein Code-Fragment bereitstellt. Das **Listing 4A** enthält nun in der Datei **Patching.xsl** die Regeln für die Transformation **T**, wobei wiederum die Befehle **template match =** und **apply-templates** zum Einsatz kommen. Im **Listing 5A** ist


30 dann der Inhalt der Datei **TestOutput.xjava(*)** mit dem modifizierten XML Quellcode und in **Listing 6A** in der Datei **TestOutput.java(*)** der modifizierte Java Quellcode dargestellt. Bei diesem Beispiel wird in der öffentlichen Testklasse die Stringzuweisung **String m_sWelcome = "hello";**

35 durch eine Stringzuweisung **String m_sWelcome = System.getProperty("WELCOME");** ersetzt, wobei hier also der feste Wert "hello" durch die vom System angeforderte

Eigenschaft "WELCOME" ersetzt wird, und die beispielsweise irrtümlich „hardcodierte“ Wertzuweisung nun „gepatched“ werden kann.

5 **Zeichnung 4** betrifft ein drittes Anwendungsbeispiel der Erfindung, bei dem die Informationen **INFO** von **Zeichnung 1** nicht nur in der oben angegebenen Weise in Form von Informationen **INFO1**, sondern auch zusätzlich Form von in den Transformationsregeln eingebetteten Informationen **INFO2** bzw.
10 Fragmenten hinzugemischt werden.

 **Zeichnung 5** betrifft ein viertes Anwendungsbeispiel der Erfindung, bei dem XML-Quellcode **CodeML** mit dem Codefragment **CF**, das die Fremdsprachenfragmente **LF1** und **LF2** enthält, durch
15 die Transformation **T** kombiniert wird, um einen modifizierten Code **CodeML***, beispielsweise angepasst an die natürliche Sprache des Benutzers (**L1=german**), zu erhalten. Die Transformation **XSLT** wird hierbei von Transformationsregeln **TR** bestimmt, die die zu ändernden Stellen des Quellcodes sowie
20 die jeweilige ausgewählte natürliche Sprache, also beispielsweise **german** oder **english**, spezifiziert. Durch das erfindungsgemäße Verfahren wird so eine Lokalisierung und Internationalisierung des Quellcodes auf effiziente und ökonomische Weise, bei einer Minimierung der hierfür erforderlichen zusätzlichen Laufzeit, erreicht.

 Die oben genannten Ausprägungen des erfindungsgemäßen Verfahrens können einzeln und in beliebiger Reihenfolge nacheinander erfolgen.

30 Durch das erfindungsgemäße Verfahren ergeben sich eine Reihe von Vorteilen, wie beispielsweise:

1. Es können schnelle und flexible Änderungen im Quellcode
35 vorgenommen werden.

2. Es ist nur ein System für Problemstellungen wie Patching, Customizing, Updating, etc. erforderlich und nicht eine Reihe verschiedener teilweise proprietärer Werkzeuge.
- 5 3. Das Verfahren basiert auf Standards wie XML und XSLT und ist hinsichtlich der Konvertierbarkeit in andere Programmiersprachen geringeren Beschränkungen unterworfen als andere Verfahren zur Modifikation von Quellcode.
- 10 4. Selbst für spezielle und komplizierte Quellcode-Modifikationen sind keine proprietären Speziallösungen erforderlich, sondern es können hierfür existierende Standards wie **XSLT**, **XPath** und **XQuery** genutzt werden.
- 15 5. Diese Art der Modifikation erlaubt die Erstellung von Hierarchien u.a. durch die Möglichkeit zur geordneten, automatisierten Hintereinanderausführung (Pipelines) mehrerer Transformationen, bspw. von Patches.
- 20 7. Die Transformationen können für eine allgemeine Wiederverwendung in XSLT-Dateien gespeichert werden, so daß Bibliotheken z.B. für bestimmte Abläufe entstehen können.
8. Es kann eine XML-Repräsentation des Quellcodes in einer XML-Datenbasis gespeichert und bei Bedarf mit Hilfe einer XSTL in einfacher Weise an die jeweiligen Kundenbedürfnisse angepasst werden (Customization).
9. Durch die Verwendung der Standards **XMLSchema** oder **DTD** oder
30 entsprechende XSLTs kann der Code vorab (ohne Kompilierung), auf bestimmte Korrektheitsaspekte hin, überprüft (validiert) werden.
10. Übliche XML-Tools können zur einfachen Bearbeitung bzw.
35 Visualisierung und Bestimmung von Beziehungen im Code verwendet werden.

11. Dauerhafte XML-basierte Programmbibliotheken, die XPath-Anfragen unterstützen, können die Wiederverwendung von Code durch besseres Auffinden eines Codes bzw. von Code-Fragmenten oder Templates verbessert werden.

Anhang 1**Listing 1: TestOutput.java**

```

5  public class TestOutput
    {
        String m_sWelcome;
    }

```

Listing 2: TestOutput.xjava

```

15  <?xml version="1.0" encoding="UTF-8"?>
    <java>
        <class>
            <modifiers><public/></modifiers>
            <name>TestOutput</name>
            <block>
                <var>
                    <type><name>String</name></type>
20         <name>m_sWelcome</name>
                </var>
            </block>
        </class>
    </java>

```

Listing 3: State.xml

```

30  <?xml version="1.0" encoding="UTF-8"?>
    <state name="m_sWelcome">
        <value>"hello"</value>
    </state>

```

Listing 4: Mixing.xsl

```

40  <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <!-- *****
        *** copy template
        *****-->
    <xsl:template match="*">
        <xsl:copy><xsl:apply-templates/></xsl:copy>
    </xsl:template>
45  <!-- *****
        *** mixing template
        *****-->
    <xsl:template match="*[(name()='var')and(name='m_sWelcome')]">
        <xsl:copy>
50         <xsl:copy-of select="*" />
        <a>
            <expr><xsl:value-of select="//state[@name='m_sWelcome']/value" /></expr>
        </a>
        </xsl:copy>
    </xsl:template>

```

</xsl:stylesheet>

Listing 5: TestOutput.xjava (*)

```
5  <?xml version="1.0" encoding="UTF-8"?>
   <java>
     <class>
       <modifiers><public/></modifiers>
10    <name>TestOutput</name>
     <block>
       <var>
         <type><name>String</name></type>
         <name>m_sWelcome</name>
15     <a>
       <expr>"hello"</expr>
     </a>
       </var>
     </block>
   </class>
20 </java>
```

Listing 6: TestOutput.java (*)

```
25 public class TestOutput
   {
     String m_sWelcome="hello";
   }
```

Anhang 2**Listing 1A: TestOutput.java**

```

5 public class TestOutput
  {
    String m_sWelcome="hello";
  }

```

Listing 2A: TestOutput.xjava

```

5 <?xml version="1.0" encoding="UTF-8"?>
  <java>
    <class>
      <modifiers><public/></modifiers>
      <name>TestOutput</name>
      <block>
        <var>
          <type><name>String</name></type>
          <name>m_sWelcome</name>
          <a>
            <expr>"hello"</expr>
          </a>
        </var>
      </block>
    </class>
  </java>

```

Listing 3A: CodeFragment.xml

```

30 <?xml version="1.0" encoding="UTF-8"?>
  <fragment name="m_sWelcome">
    <expr>
      <paren>
        <dot><name>System</name><name>getProperty</name></dot>
        <exprs><expr>"WELCOME"</expr></exprs>
      </paren>
    </expr>
  </fragment>

```

Listing 4A: Patching.xsl

```

45 <xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <!-- *****
    *** copy template
    *****-->
50 <xsl:template match="*">
  <xsl:copy><xsl:apply-templates/></xsl:copy>
</xsl:template>
  <!-- *****
    *** patching template
    *****-->
55 <xsl:template match="*[(name()='a')and(ancestor::var/name='m_sWelcome')]">

```

```

    <xsl:copy>
      <xsl:copy-of select="//fragment[@name='m_sWelcome']/expr" />
    </xsl:copy>
  </xsl:template>
5 </xsl:stylesheet>

```

Listing 5A: TestOutput.xjava (*)

```

10 <?xml version="1.0" encoding="UTF-8"?>
    <java>
      <class>
        <modifiers><public/></modifiers>
        <name>TestOutput</name>
        <block>
15      <var>
          <type><name>String</name></type>
          <name>m_sWelcome</name>
          <a>
            <expr>
              <paren>
                <dot><name>System</name><name>getProperty</name></dot>
                <exprs><expr>"WELCOME"</expr></exprs>
              </paren>
            </expr>
25          </a>
        </var>
      </block>
    </class>
  </java>

```

Listing 6A: TestOutput.java (*)

```

public class TestOutput
{
35    String m_sWelcome=System.getProperty("WELCOME");
}

```


Patentansprüche

1. Verfahren zur Modifikation von Quellcode,
- bei dem ein in einer ersten Programmiersprache formulierter
5 Quellcode (SC) in einen in einer Meta-Auszeichnungssprache
formulierten ersten Code (CodeML) umgewandelt wird,
- bei dem eine in der Meta-Auszeichnungssprache formulierte,
den späteren Programmablauf (B*) beeinflussende Information
(INFO) durch eine Transformation (T) zu diesem ersten Code
10 ersetzend oder nichtersetzend hinzugefügt und so der
ebenfalls in der Meta-Auszeichnungssprache formulierte
zweiten Code (CodeML*) gebildet wird, wobei die
Transformation in Abhängigkeit von in einer
Transformationsbeschreibungssprache formulierten
15 Transformationsregeln (TR) erfolgt, und
- bei dem dieser zweite Code in einen in der ersten
Programmiersprache oder einer anderen Programmiersprache
formulierten zweiten Quellcode (SC*) verwandelt wird, wobei
sich der Programminhalt bzw. Programmablauf (B) des ersten
20 Quellcodes (SC) vom Programminhalt bzw. Programmablauf (B*)
des zweiten Quellcodes (SC*) unterscheidet.

2. Verfahren nach Anspruch 1,
bei dem diese Information (INFO) mindestens ein Code-Fragment
(CFb) enthält und
bei dem der zweite Quellcode dadurch gebildet wird, dass
mindestens ein im ersten Quellcode enthaltenes Code-Fragment
(CFa) mit Hilfe der Transformation durch das mindestens eine
im Fragment enthaltene Code-Fragment (CFb) ersetzt wird.
30

3. Verfahren nach Anspruch 1,
bei dem die Information (INFO) speziell Daten (D) in Form von
Initialisierungszuständen (SSDb) oder Zustandsdaten (SDa)
oder Datenbankdaten (Dc) enthalten.

4. Verfahren nach Anspruch 3,
bei die Transformationsregeln (TR) von diesen Daten (D)
beeinflusst werden.
- 5
5. Verfahren nach einem der Ansprüche 1 bis 4,
bei dem Daten (D) und/oder Code-Fragmente (CF) zusätzlich in
den Transformationsregeln eingebettet sind.
- 10
6. Verfahren nach einem der Ansprüche 1 bis 5,
bei dem von checkpoints generierte Daten mittels einer
Transformation so hinzugefügt werden, das der interne Zustand
des ursprünglichen Programms beim Neustart des Programms zur
Verfügung steht bzw. von diesem genutzt werden kann.
- 15
7. Verfahren nach einem der Ansprüche 1 bis 5,
bei dem Informationen (INFO) Updates oder Patches enthalten.
8. Verfahren nach Anspruch 1 bis 5,
20 bei dem Fragmente (LF1, LF2) Informationen zur
Internationalisierung enthalten, die der Anpassung an
unterschiedliche natürliche Sprachen dienen.
9. Verfahren nach einem der Ansprüche 1 bis 5,
bei dem Daten und/oder Code-Fragmente aus einer Bibliothek
entstammen und
auf Kunden oder Kundengruppen abgestimmte Informationen
tragen.
- 30
10. Verfahren nach einem der vorhergehenden Ansprüche,
bei dem die Programmiersprache des ersten und zweiten
Quellcodes Java und die Meta-Auszeichnungssprache XML ist und
bei dem die Transformation und der Regelbeschreibung mittels
XSLT und XSL erfolgt.

11. Anordnung zur Modifikation von Quellcode,
- bei der ein erster Konverter (CONV) derart vorhanden ist,
dass ein in einer ersten Programmiersprache formulierter
5 Quellcode (SC) in einen in einer Meta-Auszeichnungssprache
formulierten ersten Code (CodeML) umgewandelt wird,
- bei der ein Prozessor derart vorhanden ist, dass eine in
der Meta-Auszeichnungssprache formulierte, den späteren
Programmablauf (B*) beeinflussende Information (INFO) durch
10 eine Transformation (T) zu diesem ersten Code ersetzend oder
nichtersetzend hinzugefügt und so der ebenfalls in der Meta-
Auszeichnungssprache formulierte zweiten Code (CodeML*)
gebildet wird, wobei die Transformation in Abhängigkeit von
in einer Transformationsbeschreibungssprache formulierten
15 Transformationsregeln (TR) erfolgt, und
- bei der ein zweiter Konverter (RCONV) derart vorhanden ist,
dass dieser zweite Code in einen in der ersten
Programmiersprache oder einer anderen Programmiersprache
formulierten zweiten Quellcode (SC*) verwandelt wird, wobei
20 sich der Programminhalt bzw. Programmablauf (B) des ersten
Quellcodes (SC) vom Programminhalt bzw. Programmablauf (B*)
des zweiten Quellcodes (SC*) unterscheidet.

Zusammenfassung

Verfahren Anordnung zur Modifikation von Quellcode unter
Einbeziehung zusätzlicher Informationen

5

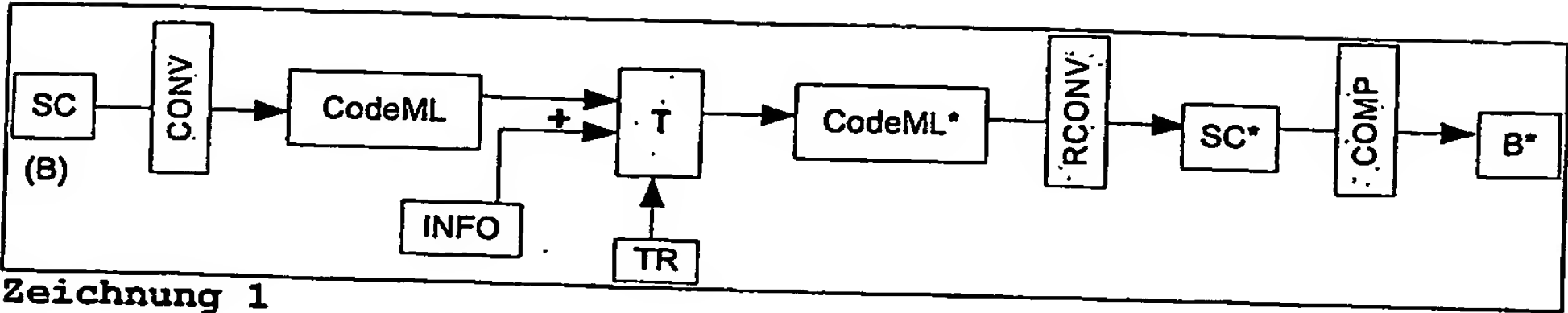
Die Erfindung besteht im Wesentlichen darin, dass der in eine
Meta-Auszeichnungssprache, beispielsweise XML, transformierte
Quellcode mit weiteren Informationen, wie beispielsweise
Ausgangszuständen, auszutauschenden Code-Fragmente und auf
die jeweilige natürliche Sprache des Anwenders zugeschnittene
Fremdsprachenmodulen, mit einer in seinen Elementen
standardisierten und übersichtlich beschreibbaren
Transformation, beispielsweise XSLT, vermischt wird, wodurch,
nach einer Rückkonvertierung aus XML in die ursprüngliche
Programmiersprache, ein neuer Quellcode entsteht, bei dem
nicht nur die Darstellung, sondern auch der eigentliche
Programminhalt bzw. der Programmablauf entsprechend den
weiteren Informationen und den Transformationsvorschriften
verändert wurde.

10

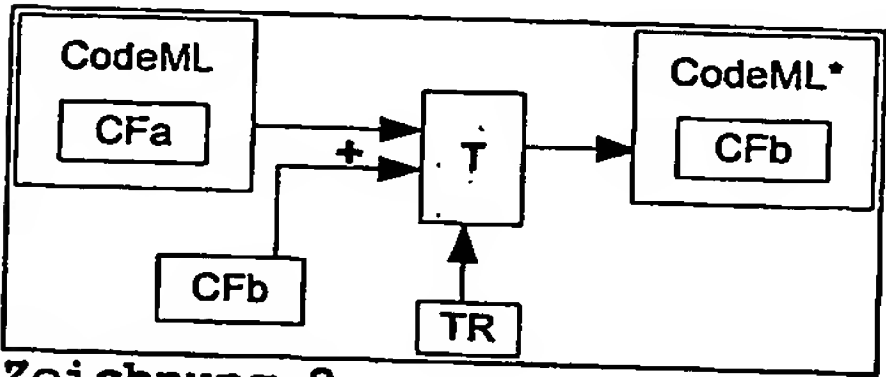
15

20

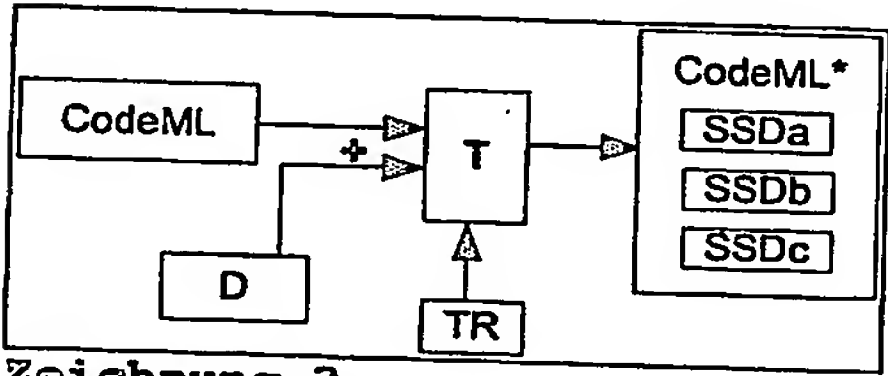
Zeichnung 1



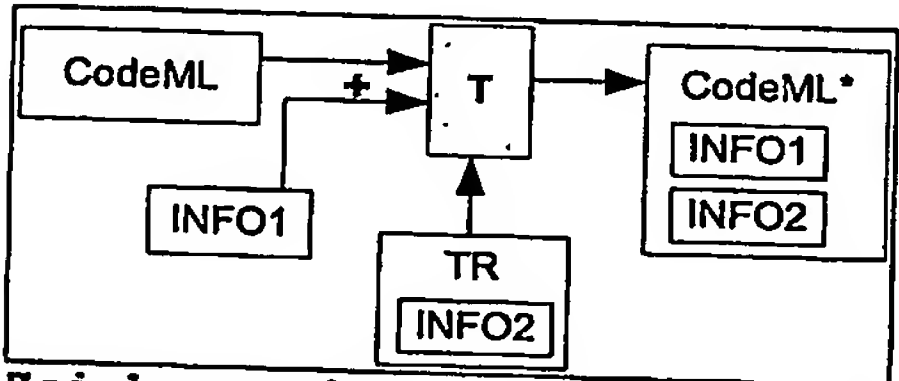
Zeichnung 1



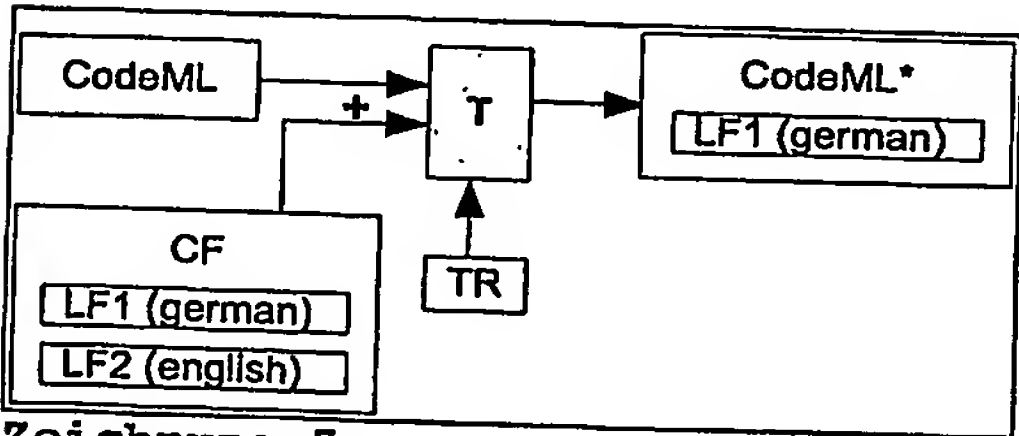
Zeichnung 2



Zeichnung 3



Zeichnung 4



Zeichnung 5